A CSP model for simple non-reversible and parallel repair plans

Carmelo Del Valle · Antonio Márquez · Irene Barba

Abstract This work presents a constraint satisfaction problem (CSP) model for the planning and scheduling of disassembly and assembly tasks when repairing or substituting faulty parts. The problem involves not only the ordering of assembly and disassembly tasks, but also the selection of them from a set of alternatives. The goal of the plan is the minimization of the total repairing time, and the model considers, apart from the durations and resources used for the assembly and disassembly tasks, the necessary delays due to the change of configuration in the machines, and to the transportation of intermediate subassemblies between different machines. The problem considers that sub-assemblies that do not contain the faulty part are nor further disassembled, but allows non-reversible and parallel repair plans. The set of all feasible repair plans are represented by an extended *And/Or* graph. This extended representation embodies all of the constraints of the problem, such as temporal and resource constraints and those related to the selection of tasks for obtaining a correct plan.

Keywords Planning · Scheduling · Constraints · Assembly · Disassembly · Repair

C. Del Valle (✉)
Depto. Lenguajes y Sistemas Informáticos,
Universidad de Sevilla, Sevilla, Spain
e-mail: carmelo@lsi.us.es

A. Márquez
Depto. Ingeniería Electrónica, Sistemas Informáticos y Automática, Universidad de Huelva, Huelva, Spain
e-mail: amarquez@uhu.es

I. Barba
Depto. Lenguajes y Sistemas Informáticos,
Universidad de Sevilla, Sevilla, Spain
e-mail: irene@lsi.us.es

Introduction

Scheduling problems have been successfully addressed for a wide scope of applications using constraint-based techniques. Most of those problems, as the resource-constrained project scheduling (RCPS) problem, can be modelled in a natural way, so that, since the actions are set, variables are chosen to correspond to the temporal unknowns (mainly start and end times) or to the ordering of tasks, and constraints gather precedence and resource constraints (Beck and Fox 1998). Some of the extensions to scheduling that have been considered, such as alternative resources and process alternatives, lead to models that are closer to planning, as problems involving choice of actions are often regarded as planning problems (Smith et al. 2000).

In the other hand, the AI planning community has done several efforts to extend classical planning techniques to treat resources and time constraints. Since real-world problems involve both planning and scheduling issues, there is an increasing interest for integrating both types of techniques (Boddy et al. 2004). Constraint programming (CP) has been used in several recent AI planners (Nareyek et al. 2005), so this paradigm is at the core for combining planning and scheduling techniques.

Some of the applications involving such issues are maintenance and repair planning, where there may be a cascading set of choices for actions, facilities, tools or personnel, which affect the duration of the plans (Smith et al. 2000). This work presents a *constraint satisfaction problem* (CSP) model for solving a planning problem corresponding to the optimal (temporal) sequencing of disassembly and assembly tasks for repairing or substituting faulty parts.

Assembly and disassembly planning are very important in the manufacturing of products and its life cycles. They involve the identification, selection and sequencing of

assembly/disassembly operations, which can be specified by their effects on the parts. The identification of assembly/disassembly operations is usually tackled by analyzing the product structure and the feasibility of each possible task (Homem de Mello and Sanderson 1991; Calton 1999), and usually leads to the set of all feasible plans. Most approaches used for choosing optimal assembly plans employ different kind of rules in order to avoid difficult tasks or awkward intermediate subassemblies (Homem de Mello and Lee 1991; Goldwasser and Motwani 1999). In other context, disassembly planning has been object of different studies, varying from maintenance or repair purposes to recycle or recovery of useful materials (Li and Zhang 1995; Lambert 1997, 1999). Different techniques have been used for solving those problems, from mathematical programming to a variety of methods related to artificial intelligence (Lambert 2003). As for other scheduling and planning problems, a CSP approach may be adequate for solving the problem proposed.

This work is focused on the selection of assembly and disassembly tasks for repairing faulty parts, and their optimal ordering. The objective is the minimization of the total reparation time when executing the plan in a generic multiple machine environment, considering different factors that can have an influence on it: durations of tasks, shared resources, including modes of operation (machine configurations), and an estimation of the time needed for doing auxiliary operations, such as the transportation of intermediate subassemblies between different machines, and set-up operations such as changes of configurations in machines. The general problem may contain different types of plans, and in this work we are concentrated on parallel and reversible repair plans that do not disassemble the subassemblies which do not include the faulty parts.

The rest of the paper is organised as follows: next section details the proposed repair planning model. Then, it is introduced the CSP model for planning the substitution or reparation of faulty parts. Later, some experimental results from different methods that solve the related problem are presented. Finally, some conclusions and future work to be developed starting from the proposed model are shown.

The repair planning problem

A usual way of describing and representing the set of all feasible assembly and disassembly plans is through *And/Or* graphs (Homem de Mello and Sanderson 1990). In these graphs, each assembly/disassembly plan is associated to an assembly/disassembly tree, an *And/Or* path starting at the root node and ending at the leaf nodes, and showing the precedence constraints among the tasks contained in the assembly/disassembly plan. In this representation, the *Or* nodes correspond to sub-assemblies, the top node corresponds

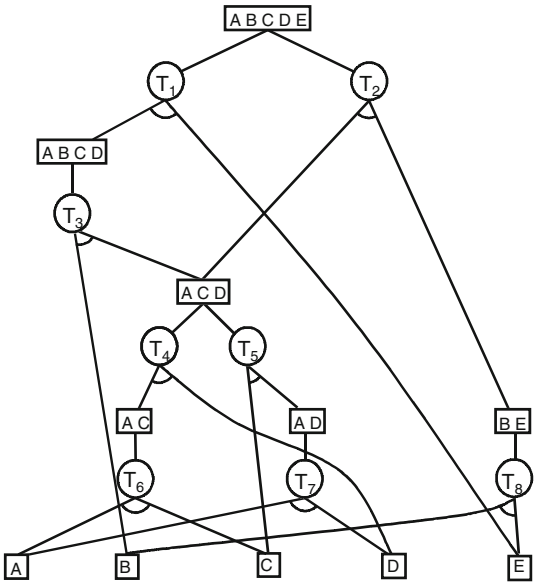

Fig. 1 The assembly *And/Or* graph for the system ABCDE

to the complete assembly, and the leaf nodes correspond to the individual parts. Each *And* node corresponds to the assembly task joining the sub-assemblies of its two nodes below it producing the sub-assembly corresponding to the node above it. The *And* nodes immediately below an *Or* node correspond to the alternative assembly tasks that might be selected in order to obtain the subassembly corresponding to the *Or* node. Furthermore, each *And* node corresponds to the disassembly task, opposite to the assembly task, that decomposes the sub-assembly above it in the representation of the *And/Or* graph (see Fig. 1). An important advantage of this representation, used in this work, is that the *And/Or* graph shows the assembly/disassembly tasks that can be executed in parallel. Furthermore, both precedence constraints and those related to the selection of tasks for obtaining a correct disassembly or assembly plan, can be easily obtained from this representation.

Considering that an assembly/disassembly task is executed in a machine with a particular configuration, other auxiliary operations are taken into account:

- set-up operations (changing the configuration of machines). $\Delta_{chf}(M, C, C')$ will denote the time needed for changing the configuration (i.e. change of tools) of the machine M from C to C'.
- transportation of subassemblies between different machines. $\Delta_{mov}(SA, M, M')$ will denote the time needed for transporting the subassembly SA from machine M to machine M'.

As explained in the next section, an extension of this representation will allow a direct mapping from the plan-

ning problem to a constraint satisfaction problem (CSP), in order to be solved using constraint programming (CP).

In order to repair or substitute a (previously detected) faulty part, a sequence of disassembly tasks must be executed before the faulty part can be extracted. After that, a repair action would substitute or repair the part, and then a set of assembly tasks must reassemble the system. In the plan, the two types of sequence-dependent auxiliary operations pointed out before (set-up machines and transportation of sub-assemblies) might have to be used.

From the AI planning perspective, the resulting planning domain would have the following durative actions:

- `assemble(sub1, sub2, result)`: assembles the subassemblies `sub1` and `sub2` to obtain a bigger subassembly `result`.
- `disassemble(result, sub1, sub2)`: disassembles the subassembly `result` to obtain two subassemblies, `sub1` and `sub2`.
- `move-subassembly(sub, mach1, mach2)`: moves the subassembly `sub` from machine `mach1` to machine `mach2`.
- `change-configuration(mach, conf1, conf2)`: changes the configuration of the machine `mach` from `conf1` to `conf2`.
- `repair-part(p)`: repairs or substitutes the part p.

The planner must obtain the optimal sequence of the disassembly operations to extract the faulty part, the replacement or repair task, and the reassembly tasks.

The general problem may involve the search in the whole *And/Or* graph, allowing different types of plans. Usually, some different properties are fulfilled, and considering them can simplify solving the problem. Some of them are taken into account in this work, as gathered in the following definitions.

Definition 1 A *repair graph* is a sub-graph of the *And/Or* graph which only contains the assembly and disassembly tasks (and the corresponding subassemblies) that could be necessary to repair some parts, according to the simplified model considered.

Definition 2 An assembly (disassembly) task T is *reversible* if its corresponding disassembly (assembly) task T' is feasible, i.e., if both tasks handle the same subassemblies, but in an opposite way, maybe using different machines or configurations. The two tasks are called *reverse* with respect to the other.

Definition 3 A *reversible plan* is a tree of the repair graph that only contains reversible tasks, so that for each disassembly task, its reverse assembly task is included.

The planning model developed in the current work supposes the conditions:

(C1) All tasks are reversible.
(C2) Subassemblies that do not include the faulty parts are not disassembled.

Some observations must be done from the conditions considered:

- Condition (C1) does not imply that plans must be reversible.
- If only one component must be repaired, condition (C1) ensures that, when (C2) is imposed, there is at least one solution, corresponding to a reversible plan where the disassembly and assembly tasks are linearly sequenced.
- In the assembly process, other subassemblies different from the ones generated in the disassembly process can appear, depending on how they are joined. That is the case when, for a disassembly task selected, its reverse assembly task is not selected.
- Disassembly tasks only handle subassemblies that contain the faulty part, whereas assembly tasks handle subassemblies that may contain or not the faulty part.
- In general, plans are not a linear sequence of tasks, unlike reversible plans. Although the disassembly process is linear, the assembly process can contain tasks that may execute in parallel with others. Moreover, it is possible that the assembly process starts before the disassembly process has finished, using those subassemblies or individual parts that are generated as disassembly tasks are executed. Additionally, although the disassembly and assembly processes are linear, there may be a parallel execution of the two types of tasks.

The CSP model

According to the problem stated in the previous section, the time and resource constraints typical from scheduling would be modified to conditional constraints taking into account that tasks (and subassemblies) may not appear in the solution. This is a similar approach found in previous works for planning and configuration applications, using DCSP models (Mittal and Falkenhainer 1990) and compiling them to standard CSPs (Do and Kambhampati 2001).

Most of the ideas are taken from previous works (Márquez et al. 2005), but the assumptions considered in this work will result in modifying most constraints and in adding others.

In order to obtain a solution, taking into account the conditions (C1) and (C2) from the previous section, the *And/Or* graph can be simplified (see Figs. 1, 2), removing those *And* nodes (disassembly) below the *Or* nodes corresponding to

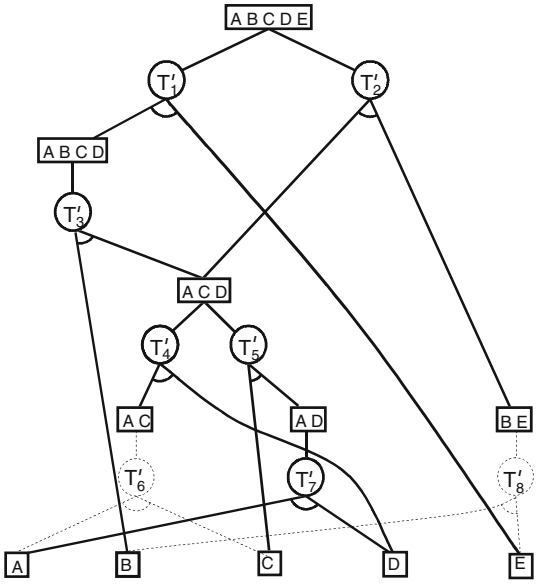

Fig. 2 The simplified disassembly *And/Or* graph for the system ABCDE when substituting part D

subassemblies which do not contain the faulty part, but maintaining the same *And* nodes for assembly which could be used in the assembly process. This can be done through a depth-first traversal of the assembly *And/Or* graph. Now, the leaf nodes of the *And/Or* assembly graph are the individual parts and the subassemblies which do not contain the faulty part. All these nodes will have the same processing, except for the faulty part. Figure 3 shows an example of this representation, where both the disassembly and assembly processes are represented. For the sake of clarity, *Or* nodes corresponding to the same subassemblies are repeated in both the disassembly and assembly parts of the graph.

Table 1 shows the number of *Or* and *And* nodes in the *And/Or* graphs corresponding to a set of hypothetical products of 30 and 40 parts. Supposing that each individual part must be repaired, it includes the average number of *Or*, *And* (assembly tasks), *And'* (disassembly tasks) nodes, and of disassembly and repair plans in the simplified *And/Or* graphs, respectively.

Variables of the CSP

Each node of the *And/Or* graph is associated to a set of variables in the CSP:

- For the assembly and disassembly tasks corresponding to each *And* node, T and T', respectively: its durations $dur(T)$ and$dur(T')$; the machines used $M(T)$ and $M(T')$, and the necessary configuration on them, $C(T)$ and $C(T')$; its starting times, $t_i(T)$ and $t_i(T')$; its ending times, $t_f(T)$ and $t_f(T')$; and two boolean variables representing if the tasks are selected for the solution, $s(T)$ and $s(T')$, respectively.
- For each subassembly SA (*Or* node): the machine used for its assembly, $m(SA)$; the machine where it is obtained after the corresponding disassembly task, $m'(SA)$; the times when it is obtained after assembly, $t_{OR}(SA)$, and disassembly, $t'_{OR}(SA)$; and two boolean variables representing if the subassembly SA appears in the assembly and disassembly processes, $s(SA)$ and $s'(SA)$, respectively.
- Finally, a part P to be repaired is associated to a temporal delay $\Delta_{subst}(P)$, corresponding to the reparation or substitution of the faulty part. In this work, we will suppose that $\Delta_{subst}(P)$ does not depend on the machine where P is extracted and/or repaired.

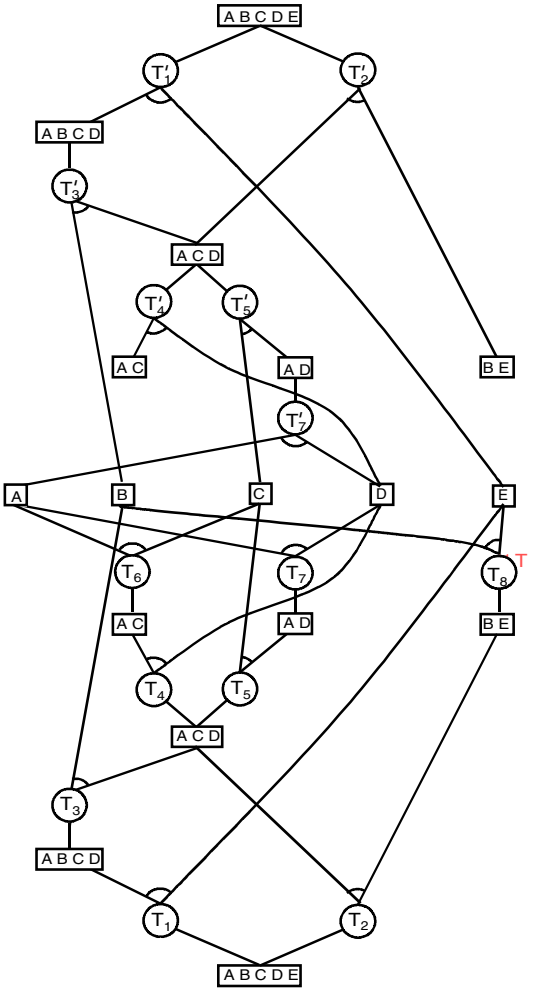

Fig. 3 The simplified repair *And/Or* graph for the system ABCDE when substituting part D

Regarding the types of actions (operators) from the AI planning perspective from section "The repair planning program", it must be noted that operators `assembly` and `disassembly` are related to the *And* nodes. The `repair-part`

Table 1 Number of *And* and *Or* nodes and plans (average) for 8 hypothetical problems

Problem	*And/Or* graph		Simplified *And/Or* graph				
	#Or	#And	#Or	#And	#And′	#Disassembly plans	#Repair plans
30a	348	630	223	327	240	623	1213
30b	404	828	303	520	365	3045	9200
30c	415	863	310	546	384	3634	12846
30d	408	837	294	506	365	3071	9414
40a	649	1518	433	833	575	9370	23005
40b	759	2086	604	1437	947	76171	405661
40c	770	2143	621	1489	984	70980	248408
40d	756	2060	598	1400	925	54449	197551

operator is associated to the *Or* node corresponding to the part P to be repaired, and the delay $\Delta_{subst}(P)$ that represents the duration of the repair task.

The extended *And/Or* graph

Although the *And/Or* graph representation shows both precedence constraints and those related to the selection of tasks for obtaining a correct disassembly and assembly plan, we extend it so that the new representation includes all the constraints involved in the problem, adding new types of links between *And* nodes. The new links represent non-precedence constraints:

- due to the use of shared resources by the tasks (constraints from group 6 below),
- taking into account the delays due to the change of configurations in the machines (constraints from group 4 below); $\Delta_{cht}(M, C, C')$ will denote the time needed for changing the configuration (for instance, change of tools) of the machine M from C to C',

The transportation of intermediate subassemblies will result in an additional delay that must be considered in the precedence constraints; $\Delta_{mov}(SA, M, M')$ will denote the time needed for transporting the subassembly SA from machine M to machine M'.

Regarding the operators from the AI planning perspective from section "The repair planning program", the operator `move-subassembly` has not an explicit task in the CSP model, and it is considered through the delays $\Delta_{mov}(\cdot)$. Similarly, the operator `change-configuration` has not an explicit task in the CSP model, and it is considered through the delays $\Delta_{cht}(\cdot)$.

Figure 4 shows the extended and simplified repair *And/Or* graph resulting for the product *ABCDE* when substituting part *D* in the general case that disassembly and assembly plans could be different. In order to understand more clearly these kinds of constraints, Fig. 5 shows only one of the possible disassembly plans of the repair *And/Or* graph resulting

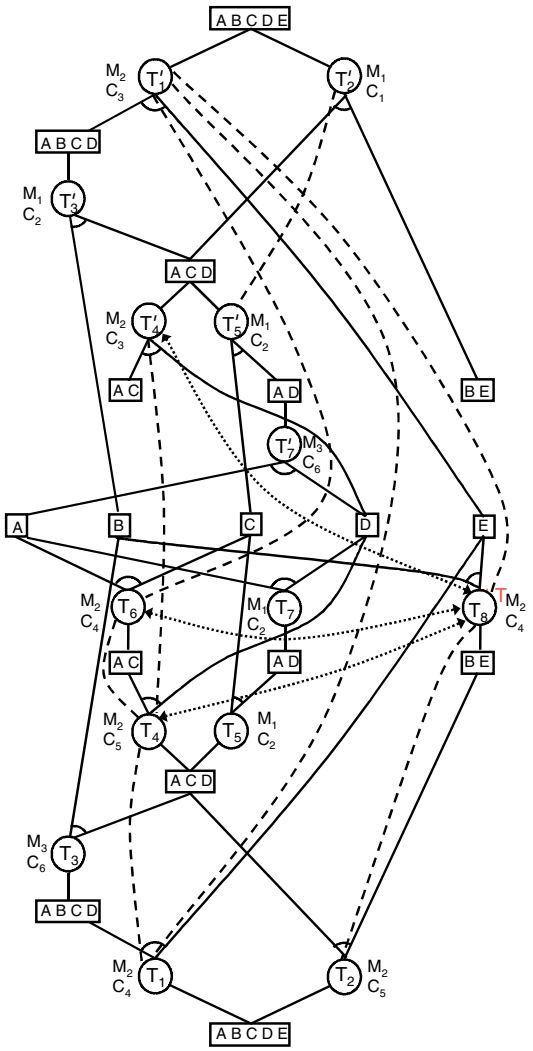

Fig. 4 The extended and simplified repair *And/Or* graph for the substitution of part D in the product ABCDE when considering all possible disassembly plans

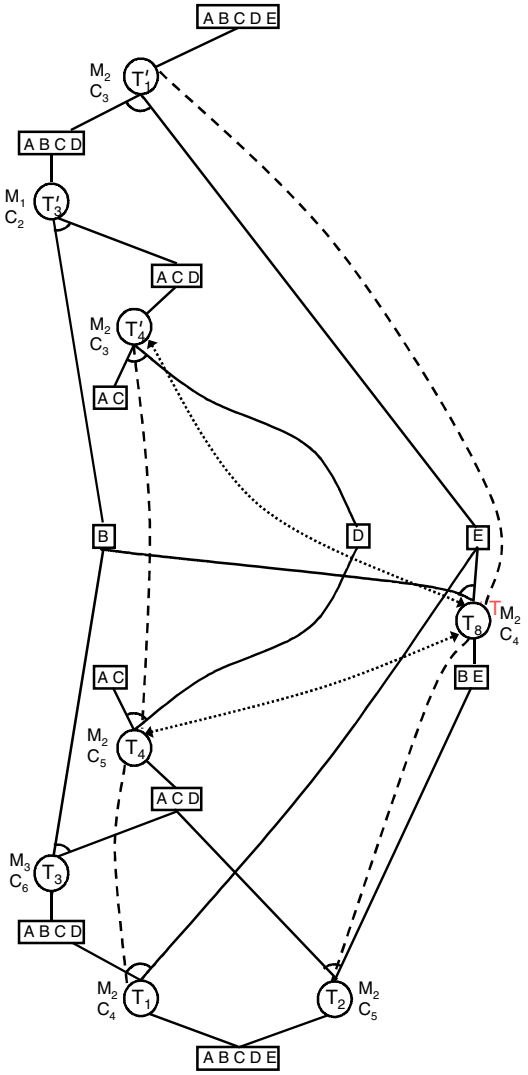

Fig. 5 The extended and simplified repair *And/Or* graph for the substitution of part D in the product ABCDE when considering only one of the disassembly plans

for the product *ABCDE* when substituting the part D, and the possible assembly plans that could complete the solution. The machines and configurations have been selected in order to describe the different cases that could appear.

Figure 5 shows that for a particular disassembly plan ($T'_1 - T'_3 - T'_4$ in the example) there may be different alternative assembly plans ($T_4 - T_3 - T_1$ and $T_4 - T_8 - T_2$ in the example). Although all the leaf nodes (AC, B, D, E) generated in the disassembly process must be present in the assembly part of the solution, the same is not true for the intermediate subassemblies, which will appear or not depending on the assembly tasks selected.

Moreover, reverse tasks may use different machines and/or configurations, so that we must take into account the possible delays due to the transportation of intermediate subassemblies between machines when reverse tasks use different machines, and the possible delays due to the change of configurations when they use the same machine.

Groups of constraints

In this subsection, the different types of constraints of the CSP for the model proposed are presented in different groups. Each group of constraints corresponds to a link or component of the extended *And/Or* graph (see Fig. 4). Some examples of them are shown in Tables 2, 3, 4, 5, and 6. Figure 6 shows the different types of graph links for each group of constraints.

Constraints from group (1) collect the relation between the information from an *Or* node and the *And* nodes below it in the original *And/Or* graph. On the one hand, they include the relation between the selection of disassembly tasks T' and assembly tasks T, and that of the sub-assembly, expressed through the XOR operator, since one and only one alternative (disassembly and assembly) task can be selected to build or disassemble a sub-assembly, if that sub-assembly is part

Table 2 Set of constraints from group (1) for the repair *And/Or* graph of Fig. 4

Group	Constraints
(1)	$s'(ABCDE) = s(ABCDE) = s'(D) = s(D)$
	$= true \wedge t'_{OR}(ABCDE) = 0$
	$\left[s'(ABCDE) \Leftrightarrow\right] \left(s(T'_1)\ XOR\ s(T'_2)\right)$
	$[s(ABCDE) \Leftrightarrow]\ (s(T_1)\ XOR\ s(T_2))$
	$s(T'_1) \Rightarrow t_i(T'_1) \geq\ t'_{OR}(ABCDE)$
	$+ \Delta_{mov}(m'(ABCDE), M(T'_1))$
	$s(T'_2) \Rightarrow t_i(T'_2) \geq\ t'_{OR}(ABCDE)$
	$+ \Delta_{mov}(m'(ABCDE), M(T'_2))$
	$s(T_1) \Rightarrow \left(t_f(T_1) = t_{OR}(ABCDE) \wedge m(ABCDE)\right.$
	$= M(T_1) = M_2)$
	$s(T_2) \Rightarrow \left(t_f(T_2) = t_{OR}(ABCDE) \wedge m(ABCDE)\right.$
	$= M(T_2) = M_2)$
	$s'(ABCD) \Leftrightarrow s(T'_3)$
	$s(ABCD) \Leftrightarrow s(T_3)$
	$s(T'_3) \Rightarrow t_i(T'_3) \geq\ t'_{OR}(ABCD)$
	$+ \Delta_{mov}\ \left(m'(ABCD), M(T'_3)\right)$
	$s(T_3) \Rightarrow t_i(T_3) = t_{OR}(ABCD)$
	$+ \Delta_{mov}\ (m(ABCD), M(T_3))$
	$s'(SA) \Rightarrow \left(s(SA) \wedge m(SA) = m'(SA) \wedge t_{OR}(SA)\right.$
	$= t'_{OR}(SA)\left.\right), SA \in \{A, B, C, E, AC, BE\}$
	$s(T_6) \Rightarrow (s(AC) \wedge m(AC) = M(T_6)$
	$= M_2 \wedge t_{OR}(AC) = t_f(T_6))$
	$\vdots$
	$\left[s'(D) \Rightarrow\right] \left(m(D) = m'(D) \wedge t_{OR}(D)\right.$
	$= t'_{OR}(D) + \Delta_{sust}(D)\left.\right)$

Table 3 Set of constraints from groups (2) and (3) for the repair *And/Or* graph of Fig. 4

Group	Constraints
(2)	$s(T_i') \Rightarrow t_f(T_i') = t_i(T_i') + dur(T_i'), \quad T_i' \in \{T_1', T_2', T_3', T_4', T_5', T_7'\}$ $s(T_i) \Rightarrow t_f(T_i) = t_i(T_i) + dur(T_i), \quad T_i \in \{T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8\}$
(3)	$s(T_1') \Rightarrow \left(s'(ABCD) \wedge s'(E)\right)$ $s(T_1) \Rightarrow (s(ABCD) \wedge s(E))$ $s(T_1') \Rightarrow \left(t'_{OR}(ABCD) = t'_{OR}(E) = t_f(T_1')\right)$ $s(T_1') \Rightarrow \left(m'(ABCD) = m'(E) = M(T_1')\right)$ $s(T_1) \Rightarrow t_i(T_1) \geq t_{OR}(ABCD) + \Delta_{mov}(ABCD, m(ABCD), M(T_1))$ $s(T_1) \Rightarrow t_i(T_1) \geq t_{OR}(E) + \Delta_{mov}(E, m(E), M(T_1))$ $\vdots$

Table 4 Set of constraints from group (4) for the repair *And/Or* graph of Fig. 4

Group	Constraints
(4)	$s'(ACD) \Leftrightarrow \left(s(T_3') \; XOR \; s(T_2')\right)$ $s'(D) \Leftrightarrow \left(s(T_4') \; XOR \; s(T_7')\right)$ $s(A) \Leftrightarrow (s(T_6) \; XOR \; s(T_7))$ $s(B) \Leftrightarrow (s(T_3) \; XOR \; s(T_8))$ $s(C) \Leftrightarrow (s(T_5) \; XOR \; s(T_6))$ $s(D) \Leftrightarrow (s(T_4) \; XOR \; s(T_7))$ $s(E) \Leftrightarrow (s(T_1) \; XOR \; s(T_8))$ $s(ACD) \Leftrightarrow (s(T_3) \; XOR \; s(T_2))$

Table 5 Set of the constraints from group (5) for the repair *And/Or* graph of Fig. 4

Group	Constraints
(5)	$\left(s(T_1') \wedge s(T_1)\right) \Rightarrow t_i(T_1) \geq t_f(T_1') + \Delta_{cht}(M_2, C_3, C_4)$ $\left(s(T_1') \wedge s(T_6)\right) \Rightarrow t_i(T_6) \geq t_f(T_1') + \Delta_{cht}(M_2, C_3, C_4)$ $\left(s(T_1') \wedge s(T_8)\right) \Rightarrow t_i(T_8) \geq t_f(T_1') + \Delta_{cht}(M_2, C_3, C_4)$ $\left(s(T_2') \wedge s(T_5')\right) \Rightarrow t_i(T_5') \geq t_f(T_6') + \Delta_{cht}(M_1, C_1, C_2)$ $\left(s(T_4') \wedge s(T_4)\right) \Rightarrow t_i(T_4) \geq t_f(T_4') + \Delta_{cht}(M_2, C_3, C_5)$ $(s(T_6) \wedge s(T_4)) \Rightarrow t_i(T_4) \geq t_f(T_6) + \Delta_{cht}(M_2, C_4, C_5)$ $(s(T_8) \wedge s(T_2)) \Rightarrow t_i(T_2) \geq t_f(T_8) + \Delta_{cht}(M_2, C_4, C_5)$ $(s(T_4) \wedge s(T_1)) \Rightarrow t_i(T_1) \geq t_f(T_4) + \Delta_{cht}(M_2, C_5, C_4)$

of the plan solution:

$$s'(SA) \Leftrightarrow XOR_{T_i' \in succ(SA)}(s(T_i'))$$
$$s(SA) \Leftrightarrow XOR_{T_i \in pred(SA)}(s(T_i))$$

In these expressions, *succ*(*SA*) refers to the set of the alternative disassembly tasks which can be selected for the disassembly of *SA*, and *pred*(*SA*) refers to the set of the alternative assembly tasks which can be selected for obtain *SA* (see Fig. 6).

Table 6 Set of constraints from group (6) for the repair *And/Or* graph from Fig. 4

Group	Constraints
(6)	$\left(s(T_4') \wedge s(T_8)\right) \Rightarrow$ $(t_i(T_4') \geq t_f(T_8) + \Delta_{cht}(M_2, C_4, C_3) \vee t_i(T_8)$ $\geq t_f(T_4') + \Delta_{cht}(M_2, C_3, C_4))$ $(s(T_6) \wedge s(T_8)) \Rightarrow \left(t_i(T_6) \geq t_f(T_8) \vee t_i(T_8) \geq t_f(T_6)\right)$ $(s(T_4) \wedge s(T_8)) \Rightarrow$ $(t_i(T_4) \geq t_f(T_8) + \Delta_{cht}(M_2, C_4, C_5) \vee t_i(T_8)$ $\geq t_f(T_4) + \Delta_{cht}(M_2, C_5, C_4))$

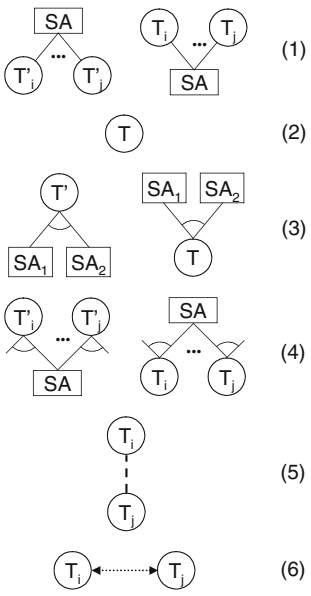

Fig. 6 Types of graph links for groups of constraints

On the other hand, these constraints allows to establish the disassembly times t'_{OR} and assembly times t_{OR} of *Or* nodes related to the start times of the disassembly tasks or the end

times of the assembly tasks, respectively:

$$s(T_i') \Rightarrow t_i(T_i') \geq t'_{OR}(SA) + \Delta_{mov}\left(SA, m'(SA), M(T_i')\right)$$
$$s(T_i) \Rightarrow t_f(T_i) = t_{OR}(SA)$$

and the machine m where a sub-assembly is generated after an assembly task:

$$s(T_i) \Rightarrow m(SA) = m(T_i)$$

A special case is for the complete product and for the faulty part, which always will appear in the solution, so that the corresponding boolean variables s are *true*. Moreover, for the *Or* leaf nodes (individual parts and sub-assemblies that do not include the faulty part) t'_{OR} and t_{OR} are set equals, except for the faulty part, where the delay corresponding to the reparation is considered. The origin of times is set to the t'_{OR} variable of the complete product, and the goal is the minimization of the t_{OR} variable of the complete product. Table 2 shows some representative examples of the different constraints from group (1) corresponding to the extended and simplified repair *And/Or* graph of Fig. 4.

Constraints from group (2) consider the durations of assembly and disassembly tasks and correspond to the relationships between the starting and ending times of the assembly and disassembly tasks. So, for any assembly or disassembly task T:

$$s(T) \Rightarrow t_f(T) = t_i(T) + dur(T)$$

Constraints from group (3) (see Fig. 6) collect the relation between the information from an *And* node and the (two) *Or* nodes below it in the original *And/Or* graph. Apart from the obligatory selection of the two *Or* nodes if the *And* node is selected, i.e.

$$s(T') \Rightarrow s'(SA_1) \wedge s'(SA_2)$$
$$s(T) \Rightarrow s(SA_1) \wedge s(SA_2)$$

they also include the equality constraint between the disassembly times of the *Or* nodes, t'_{OR}, and the end time of a disassembly task T' above them in the original *And/Or* graph,

$$s(T') \Rightarrow t_f(T') = t'_{OR}(SA_1) = t'_{OR}(SA_2)$$

and the precedence between the assembly time of the *Or* nodes t_{OR} and the start times of assembly task T (*And* nodes), and considering the possible delays due to the transportation of sub-assemblies if the two consecutive (disassembly or assembly) tasks involving it use different machines:

$$s(T) \Rightarrow t_i(T) \geq t_{OR}(SA_1) + \Delta_{mov}\left(SA, m(SA_1), M(T)\right)$$
$$s(T) \Rightarrow t_i(T) \geq t_{OR}(SA_2) + \Delta_{mov}\left(SA, m(SA_2), M(T)\right)$$

Moreover, the machine m' where a sub-assembly is generated after a disassembly task is the machine used by the disassembly task:

$$s(T') \Rightarrow m'(SA_1) = M(T')$$
$$s(T') \Rightarrow m'(SA_2) = M(T')$$

Table 3 shows some indicative examples of the different constraints from groups (2) and (3) corresponding to the extended and simplified repair *And/Or* graph of Fig. 4.

Constraints from group (4) collect the relation between the selection of an *Or* node and that of all the *And* nodes above it (possibly only one) in the original *And/Or* graph. The temporal constraints between those nodes are included in the group of constraints (3) depicted before. Figure 6 shows that the disassembly tasks T_i' involved are the immediate predecessors of the sub-assembly, SA, and that the assembly tasks T_i involved are the immediate successors of SA. So, the general form of the constraints is

$$s'(SA) \Leftrightarrow XOR_{T_i' \in pred(SA)}\left(s(T_i')\right)$$
$$s(SA) \Leftrightarrow XOR_{T_i \in succ(SA)}\left(s(T_i)\right)$$

Table 4 shows some indicative examples of the different constraints from group (4) corresponding to the extended and simplified repair *And/Or* graph of Fig. 4.

All the previous types of constraints come from the relations between the nodes included in the original *And/Or* graph. The next two groups of constraints come from the use of (same or different) resources by the different assembly and disassembly tasks, and they are related to new links between tasks in the extended *And/Or* graph.

Constraints from group (5) are due to the delay needed for a change of configuration in a machine between the executions of assembly or disassembly tasks using the same machine with precedence constraints among them. Those constraints may include the relations between reverse disassembly and assembly tasks. Notice that, for a particular repair plan, it is only needed to relate each task to its closest successor task, in any possible subsequence of tasks, that uses the same machine. For a task T_i, and its closest successor task T_j that uses the same machine m, taking into account the possible change of configuration in the machine, the constraint

$$\left(s(T_i) \wedge s(T_j)\right) \Rightarrow t_i(T_j) \geq t_f(T_i) + \Delta_{cht}\left(m, C(T_i), C(T_j)\right)$$

must be satisfied. Notice that, when both tasks use the same configuration, the resulting constraint is superfluous and can be eliminated. Moreover, since the solution may contain non-reverse tasks, each disassembly task must be related to each closest (maybe not only one) successor assembly task that uses the same machine. For the example of Fig. 5, the disassembly task T_1' is related to T_8 and, as T_1' and T_4 use the same

configuration, the corresponding constraint is not necessary. Also, the relation between T_1' and T_1, present in the example of Fig. 4 because of the possible plan $T_1' - T_3' - T_5' - T_7' - T_7 - T_5 - T_3 - T_1$, is not necessary for the example of Fig. 5, because if T_1 is selected, it would be executed after T_4' and T_4, both of them executed after T_1'.

Table 5 shows some indicative examples of the different constraints from group (5) corresponding to the extended and simplified repair *And/Or* graph of Fig. 4.

Finally, constraints from group (6) take into account that some (assembly or disassembly) tasks may execute in parallel depending on the use of shared resources. For each two tasks T_i and T_j requiring the same machine m, with no precedence constraint among them, and which may belong to the same repair plan, these constraints express the two possible orders of execution of the tasks:

$$\begin{aligned}&\big(s(T_i) \wedge s(T_j)\big) \Rightarrow \big(t_i(T_i) \geq t_f(T_j)\\ &\quad +\Delta_{cht}\left(m, C(T_j), C(T_i)\right) \vee t_i(T_j) \geq t_f(T_i)\\ &\quad +\Delta_{cht}\left(m, C(T_i), C(T_j)\right)\big)\end{aligned}$$

For the example of Fig. 4, the assembly task T_8 is related to the disassembly task T_4' and to the assembly task T_4 in the repair plan $T_1' - T_3' - T_4' - T_8 - T_4 - T_2$, showing that assembly task T_8 may be executed before or after T_4' and T_4. Table 6 shows some indicative examples of the different constraints from group (6) corresponding to the extended and simplified repair *And/Or* graph of Fig. 4.

A typical objective for such a problem would be the minimization of the elapsed time of the plan (makespan), that is, the time when the system is reassembled after the reparation, that is given by the variable t_{OR} (ABCDE) for the example used.

Notice that the combinatorial character of the problem is due to the XOR constraints from groups (1) and (4) and the disjunctive constraints from group (6). These types of constraints correspond, respectively, to the selection of alternative (assembly and/or disassembly) tasks and to the use of shared resources by the assembly and/or disassembly tasks that are not related through precedence constraints.

Experimental results

In this section some experimental results related to different algorithmic methods for obtaining repair plans are shown. The CSP model described in this paper has been tested using a basic backtracking-based algorithm implemented in ILOG Solver (ILOG). We refer it as ALG-2 in the rest of the paper. A temporal limit of 300 s was established for the search. In order to guide the search, the order of selection of variables to be instantiated is from up to down in the extended *And/Or* graph (Fig. 4).

Table 7 Comparative results (quality of solutions)

Problem	Best			Δ (Best)			# Opt
	SGPlan	ALG-1	ALG-2	SGPlan	ALG-1	ALG-2	
30a	0, 012	0, 037	1	0, 498	0, 083	0	58
30b	0, 050	0, 562	0, 425	0, 360	0, 058	0, 915	0
30c	0, 050	1	0	0, 418	0	2, 266	0
30d	0, 075	1	0	0, 382	0	1, 367	0
40a	0, 087	1	0	0, 368	0	1, 306	0
40b	0, 150	0, 450	0, 550	0, 449	0, 066	0, 913	44
40c	0, 012	0, 212	0, 850	0, 496	0, 082	0, 018	44
40d	0, 137	1	0	0, 350	0	0, 529	0

Table 8 Comparative results (computation time)

Problem	Time Ave.		
	SGPlan	ALG-1	ALG-2
30a	0,193	0,044	120, 9
30b	0,485	0,097	300
30c	0,604	0,091	300
30d	0,598	0,108	300
40a	1,395	0,157	300
40b	2,063	0,264	136, 1
40c	1,407	0,193	136, 2
40d	1,788	0,275	300

Tables 7 and 8 show a comparison of three different algorithms used to solve the repair planning problem. Each row refers to a set of 80 instances of an *And/Or* graph for a hypothetical product of 30 or 40 parts, with different combinations for the durations of tasks, machines and configurations used, and faulty part selected to be repaired. The experiments were carried out on a 2,13 GHz Intel Core 2 Duo with 2 GB RAM.

First of all, a generic planner, SGPlan (Chen et al. 2006), winner of the 1st Prize Satisficing (sub-optimal) Planning in the Deterministic Part of the International Planning Competition (IPC-5) in 2006 (Dimopoulos et al. 2006), has been used to solve the corresponding problems related to an adapted domain definition.

The second algorithm used was chosen from a previous work (Márquez et al. 2005). We refer it as ALG-1 in the rest of the paper. This algorithm, implemented in ILOG Solver and Scheduler (ILOG), uses a dynamic CSP approach, and a more restricted model that only allows to obtain linear plans.

According to the three algorithms previously mentioned, Tables 7 and 8 give results on the quality of solutions and the computation times, showing the following measures:

- Best: fraction of solutions for each method with the same makespan than the best solution obtained by the three algorithms.

- Δ Best: average deviation related to the best solution found.
- # opt: number of proven optimal solutions found by ALG-2 (from the 80 instances per row).
- Time Ave.: average time spent by the algorithm to obtain the solution.

The results from Table 7 show that the behaviour of ALG-2 is highly dependent of the graph structure. In some cases (30a and 40c) this algorithm obtains the best solutions, but in other cases (30c, 30d, 40a and 40d) the other algorithms are better. In other cases (30b and 40b) some of the problems are solved in the best way and others in the worst one. Also, the optimal solution is guarantied in some kinds of problems (30a, 40b and 40c), but in others, the mean deviation from the best solution was very high.

The results obtained by ALG-1 are the best in some cases (30c, 30d, 40a and 40d) and obtained the second place in the other cases. The mean deviation from the best for ALG-1 was the better in general, because it carries out a complete search but for the optimal linear plan. On the other hand, the results of SGPlan are the worst in most cases regarding the number of best solutions found.

The computational times given in Table 8 show that the fastest algorithm is ALG-1 and the slowest one is ALG-2, as expected for such a basic algorithm.

Conclusions and future work

This work proposes a CSP model for the planning and optimal sequencing of disassembly and assembly tasks when repairing or substituting faulty parts. Two assumptions are made, so that subassemblies that do not contain the faulty part are nor further disassembled, but allowing non-reversible and parallel repair plans.

The CSP can be solved directly using conventional methods for a generic CSP, since all the variables and constraints of the problem are included, as has been shown in the previous section. Also, a DCSP approach can be used, so that the selection of tasks and subassemblies would impose the addition or deletion of the consequent part of the constraints.

As future work, apart from the use of different algorithms for solving the problem, we propose the development of more general models, considering the maintenance and repairing of multiple parts, and that tasks may be not reversible, where the solution might imply to decompose subassemblies which do not include the faulty parts.

Acknowledgements This work has been partially supported by the Spanish Ministerio de Educación y Ciencia through a coordinated research project (Grant DIP2006-15476-C02-01) and Feder (ERDF).

References

Beck, J. C., & Fox, M. S. (1998). A generic framework for constraint-directed search and scheduling. *AI Magazine, 19*(4), 101–130.

Boddy, M. S., Cesta, A., & Smith, S. F., (Eds.). (2004). Workshop on "Integrating planning into scheduling" (WIPIS-04). In *14th International Conference on Automated Planning and Scheduling (ICAPS), Whisler Canada*, June, 2004. http://pst.istc.cnr.it/wipis-at-icaps-04/.

Calton, T. L. (1999). Advancing design-for-assembly. The next generation in assembly planning. In *Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning* (pp. 57–62). IEEE Catalog Number 99TH8470, ISBN 0-7803-5704-3, doi:10.1109/ISATP.1999.782935.

Chen, Y., Hsu, C., & Wah, B. (2006). Temporal planning using subgoal partitioning and resolution in SGPlan. *Journal of Artificial Intelligence Research, 26*, 323–369.

Dimopoulos, Y., Gerevini, A., Haslum, P., & Saetti, A. (2006). The fifth International Planning Competition, hosted at the International Conference on Automated Planning and Scheduling (ICAPS-06), Cumbria, UK, 2006. http://ipc5.ing.unibs.it/index.html.

Do, M. B., & Kambhampati, S. (2001). Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. *Artificial Intelligence, 132*, 151–182.

Goldwasser, M. H., & Motwani, R. (1999). Complexity measures for assembly sequences. *International Journal of Computational Geometry and Applications, 9*, 371–418.

Homem de Mello, L. S., & Lee, S. (Eds.). (1991). *Computer-aided mechanical assembly planning*. Kluwer Academic Publishers.

Homem de Mello, L. S., & Sanderson, A. C. (1990). And/Or graph representation of assembly plans. *IEEE Transactions on Robotics and Automation, 6*(2), 188–199.

Homem de Mello, L. S., & Sanderson, A. C. (1991). A correct and complete algorithm for the generation of mechanical assembly sequences. *IEEE Transactions Robotic & Automation, 7*(2), 228–240.

ILOG, France, ILOG Solver and Scheduler, http://www.ilog.com/.

Lambert, A. J. D. (1997). Optimal disassembly of complex products. *International Journal of Production Research, 35*, 2509–2523.

Lambert, A. J. D. (1999). Optimal disassembly sequence generation for combined material recycling and part reuse. In *Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning* (pp. 146–151). IEEE Catalog Number 99TH8470, ISBN 0-7803-5704-3, doi:10.1109/ISATP.1999.782950.

Lambert, A. J. D. (2003). Disassembly sequencing: a survey. *International Journal of Production Research, 41*(16), 3721–3759.

Li, W., & Zhang, C. (1995). Design for disassembly analysis for environmentally conscious design and manufacturing. In *Proceedings of the International Mechanical Engineering Congress and Exposition*, November 12–17, 1995, San Francisco, California (pp. 969–976). ISBN 0791817385.

Márquez, A., Del Valle, C., Gasca, R. M., & Toro, M. (2005). A constraint-based algorithm for planning the substitution of faulty parts. *In Frontiers in Artificial Intelligence and Applications, 117*, 79–88.

Mittal, S., & Falkenhainer, B. (1990). Dynamic constraint satisfaction problems. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 25–32). AAAI Press.

Nareyek, A., Freuder, E. C., Fourer, R., Giunchiglia, E., Goldman, R.P., Kautz, H.A., Rintanen, J., & Tate, A. (2005). Constraints and AI planning. *IEEE Intelligent Systems, 20*(2), 62–72.

Smith, D., Frank, J., & Jónsson, A. (2000). Bridging the gap between planning and scheduling. *Knowledge Engineering Review, 15*(1), 47–83.